ARMY RESEARCH LABORATORY

Modeling Genetic Regulatory Networks Using First-Order Probabilistic Logic

by Nand Kishore, Radhakrishnan Balu, and Shashi P. Karna

ARL-TR-6354 **March 2013**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5069

ARL-TR-6354 **March 2013**

Modeling Genetic Regulatory Networks Using First-Order Probabilistic Logic

Nand Kishore
Thomas Jefferson High School for Science and Technology of Virginia

Radhakrishnan Balu and Shashi P. Karna
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
March 2013	Final	16 June 2012–9 August 2012

4. TITLE AND SUBTITLE
Modeling Genetic Regulatory Networks Using First-Order Probabilistic Logic

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Nand Kishore,* Radhakrishnan Balu, and Shashi P. Karna

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-WMM-A
Aberdeen Proving Ground, MD 21005-5069

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-6354

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
*Thomas Jefferson High School for Science and Technology of Virginia

14. ABSTRACT
New technologies such as microarrays and flow cytometry have led to the availability of large amounts of biological data. There is a need to model biological systems to aid in medication and drug delivery. Genetic Regulatory Networks (GRNs) represent the signal transduction, or the activation and deactivation of genes, as their corresponding proteins directly or indirectly interact with one another. GRNs can be modeled using statistical and logical techniques, more precisely using Bayesian networks. Bayesian networks are directed acyclic graphs (DAGs) where the nodes represent random variables and edges represent conditional dependencies. In this research, a learning algorithm was implemented to determine the structure and the parameters of Bayesian networks that model GRNs from real data. PRISM, a probabilistic learning framework based on B-prolog, was used to program the Bayesian networks. Instead of conventional statistical techniques, which rely on point estimates, the method of Variational Bayes-Expectation Maximization (VB-EM) was used to obtain a lower-bound to the marginal likelihood, or the free energy, and a set of optimal parameters. A hill-climbing algorithm using this free energy as a scoring function was utilized. The learning algorithm was tested on the well-established "Raf" pathway.

15. SUBJECT TERMS
artificial intelligence, prolog, gene regulation, "Raf" pathway

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Radhakrishnan Balu
Unclassified	Unclassified	Unclassified	UU	28	19b. TELEPHONE NUMBER (*Include area code*) (410) 306-4787

Contents

List of Figures

List of Tables

INTENTIONALLY LEFT BLANK.

1. Introduction

New advances in technology have led to the growth of advanced detection techniques in molecular biology. This rise in turn has led to the availability of large amounts of biological data. There has been a growing need to model biological systems to further understand biological processes, and facilitate the development of new techniques for medication and drug delivery (*1*).

Genetic Regulatory Networks (GRNs) model the relationships between the activation and deactivation of genes. They represent the signal transduction of proteins, as proteins indirectly and directly interact to regulate the expression of certain genes. GRNs are responsible for many of the most fundamental biological processes, including T-cell production, mitosis, protein synthesis, metabolism, etc. (*2*).

Modeling GRNs is important to facilitate medication and drug delivery. Researchers can target specific genes or proteins in signal transduction pathways to cure a number of diseases and disorders, with applications in improving the immune response and treating cancer (*2*). However, this requires a detailed knowledge about the effects of activating or deactivating multiple genes in the network. Statistical and logical models are natural choices to represent the complex relationships between genes and proteins as a signal propagates through the pathway. Furthermore, modeling techniques must be flexible and provide a basis to conduct inference on the activation of certain genes.

However, it is difficult to develop models that can accurately describe the relationships between genes. Furthermore, depending on the gene expression data, a gene may not always activate given certain conditions as the activation of a gene is stochastic in nature. Therefore, to develop such complex processes one can use the paradigm of probabilistic logical programming. Probabilistic logical programming allows users to form connections between otherwise independent random variables and develop logical relationships. Furthermore, it provides a simple but powerful framework to implement logical and statistical techniques (*3*). Specifically in the case of GRNs, probabilistic logical programming allows one to implement and facilitate the structural and parameter learning of Bayesian networks, which can provide an accurate and flexible means of modeling GRNs from real gene expression data (*4*).

Proteins are one of the most important biochemical compounds in living organisms. Synthesized from genes, proteins are composed of 20 naturally occurring amino acids held together in polypeptide bonds. Proteins serve in a number of important biological processes including metabolism, where they serve as enzymes for catalyzing reactions, the immune response, where antibodies target foreign compounds for destruction, and in cell signaling (*5*).

Two major types of proteins, extracellular and trans-membrane, participate in cell signaling. Trans-membrane proteins span the cell membrane and serve as receptors for signaling molecules. Extracellular proteins carry or transmit signals across cells. When an extracellular protein hits a receptor on a cell, it induces a biochemical response inside the cell. This response propagates through signal transduction pathways, eventually leading to the activation of a gene (*6*).

It is important to recognize that each protein corresponds to a certain gene from which it is synthesized. These genes have different "expression levels" that can be quantified in terms of its concentration. There are a few major techniques to measure gene expression levels. The first is through microarrays, where microscopic deoxy ribonucleic acid (DNA) probes are placed on a slide. When another slide of tissue containing messenger ribooxy nucleic acid (mRNA) comes into contact with the slides, the mRNA hybridizes with its complementary probe, allowing one to measure the mRNA levels. A significant assumption in using these measurements is that mRNA levels are directly proportional to gene expression. Another method to measure expression levels is to use flow cytometry, which uses the principles of light scattering and light excitation to accurately measure absolute protein concentrations, as well as other properties (*4*). The assumption is made that protein activity is proportional to its concentration. Data from experiments for established GRNs are readily available from sources such as National Institute of Health.

An important aspect of GRNs is that the expression levels of one gene may be related to the expression levels of another gene, either indirectly or directly through the interactions of their corresponding proteins. When this happens, the gene is said to "regulate" the other gene (*1*). For example, when a cell receives a signal through a receptor on its membrane, it begins the process of gene activation and protein synthesis. A certain gene is transcribed into mRNA, translated into amino acids, and finally folded into a structure forming a protein. This protein then goes off to carry out other functions, which may then trigger the activation of another gene through cell signaling (*6*).

The immune response can be modeled as a series of activations or deactivations of certain proteins. The immune system functions by first monitoring the body for any pathogens or foreign material (*6*). If detected, the immune system sends a signal from the source to other cells to combat the pathogen. This signal causes the activation of a gene, which may then continue to activate another gene, and so on. Such interactions between genes or proteins are known as GRNs and more traditionally known as signal transduction pathways (*1*).

2. MAPK/Raf Pathway

The pathway that has been analyzed in this research is the MAPK/Raf pathway. This pathway aids in human cell division or mitosis, and carries a signal from outside the cell to induce protein synthesis and mitosis inside the cell. The pathway functions when some extracellular protein, known as a mitogen, hits a receptor on the cell membrane. This causes a bio-chemical response in the cell that activates certain proteins, such as PKC and PKA. The signal propagates through the pathway, hitting important proteins such as P38, JNK, Raf, and Mek. The signal eventually reaches Erk, which activates a transcription factor that activates a gene, eventually leading mitosis (*7*).

3. Bayesian Network

Given biological data, such as microarray and flow cytometry data, the problem is to develop a model that can represent the GRN as closely as possible. Basically, a model needs to be generated from the real data that can both describe the underlying structure of the network and show the effects of certain proteins activation on the entire network. As the activation of a certain protein is probabilistic in nature, the activation of a certain gene may not always activate a protein downstream. The uncertainty in the pathway given the statistical nature of the data has to be captured.

The interactions between these genes can be modeled graphically through a network where the nodes correspond to gene (or protein) names and the edges indicate relationships. More specifically, a type of network that can be used to model these complex relationships is the Bayesian network (*4*). Formally, a Bayesian network is a probabilistic directed acyclic graph (DAG) where the nodes represent random variables and the edges represent conditional dependencies. Each node contains a probability distribution that gives the conditional probability of that node activating depending on its parents. Although the relationships are statistical in nature, they can still be used to indicate causality (*2*).

In the case of GRNs, a Bayesian network can be developed where the nodes correspond to the genes (or proteins) and the edges represent conditional dependencies. The probability distribution at each node would represent the probability of that protein activating given the activation/deactivation of its parents.

Bayesian networks are attractive structures to use as they can easily model complex relationships and find the posterior probabilities given the activation of certain nodes. Furthermore, Bayesian networks appeal to the naturally probabilistic nature of signal transduction. A protein, at least

from a high-level statistical standpoint, may not always activate depending on certain conditions. By representing the uncertainty in activation, Bayesian networks are more flexible than other statistical analysis techniques.

4. Methods

The main learning problem can then be stated as creating a Bayesian network that models genetic regulatory networks from microarray or flow cytometry data. This problem essentially reduces to two sub-problems, which are (1) parameter learning and (2) structure learning. Parameter learning consists of finding the correct probability distribution for each node in the network. Structural learning, the more complicated of the two, is learning the actual structure or which edges connect to which nodes. For structural learning, each network requires a score for comparison. Since it is computationally intractable to compare each possible network, it is appealing to use artificial intelligence techniques to generate the structure.

There are a few issues in the computations required for a Bayesian network. First, the data might be incomplete with hidden or latent variables. Second, finding the likelihood exactly for the network is impossible in most cases, since the integral that arises is usually intractable. The method that can be used to solve this problem is known as Variational Bayes-Expectation Maximization (VB-EM), which can provide an approximation to the marginal likelihood for the network by using the mean field approximation. Rather than use a point estimate for the parameters, Variational Bayes treats both the parameters and any hidden variables as unknowns from a Bayesian standpoint. In the process, Variational Bayes also obtains a set of optimal parameters (*8*). The estimation to the marginal likelihood, also known as the (negative) free energy, can be used as a score for model selection. This technique is different from other methods such as ML (Maximum Likelihood) and MAP (Maximum a posteriori), which use point estimates for the parameters. Monte Carlo methods are able to take random samples and evaluate integrals at different values of the parameter, but are computationally intensive (*2*). Since the Variational Bayes method can be implemented iteratively it is a method of choice for use in learning problems (*8*).

A derivation of Variational Bayes based on Sato et al. (*9*) and Beal (*8*) is given in equation 1. It is of interest to find some posterior probability, p(M|D), where M is the model and D is the observed data. Bayes theorem says

$$p(M|D) \propto p(D|M)p(M). \quad (1)$$

So, we must then compute this likelihood, p(D|M). However, first the model is parameterized by some parameters θ. Second, there may be hidden or latent variables Z that have not been observed. Then, p(D|M) can be expressed as

$$p(D \mid M) = \sum_z \int_\Theta p(D, z, \theta \mid M) d\theta . \tag{2}$$

We must somehow evaluate this expression to compute the marginal likelihood. However, the above integral is intractable. So, instead of considering this expression, we can rather consider the log of this expression, L(D):

$$L(D) = \log \sum_z \int_\Theta p(D, z, \theta \mid M) d\theta . \tag{3}$$

We can then re-express this as

$$L(D) = \log \sum_z \int_\Theta p(D, z, \theta \mid M) d\theta = \log \sum_z \int_\Theta q(z, \theta \mid D.M) \frac{p(D, z, \theta \mid M)}{q(z, \theta \mid D, M)} d\theta , \tag{4}$$

where q(.) is the distribution of the hidden variables and the parameters given the data and model.

Finally, using this form of the equation, we can then make an appeal to Jensen's inequality:

$$\begin{aligned} L(D) &= \log \sum_z \int_\Theta p(D, z, \theta \mid M) d\theta = \log \sum_z \int_\Theta q(z, \theta \mid D, M) \frac{p(D, z, \theta \mid M)}{q(z, \theta \mid D, M)} d\theta \\ &\geq \sum_z \int_\Theta q(z, \theta \mid D, M) \log \frac{p(D, z, \theta \mid M)}{q(z, \theta \mid D, M)} d\theta . \end{aligned} \tag{5}$$

Jensen's inequality allows to put a lower bound on the marginal likelihood. The right side of this inequality is known as F(q), the (negative) variational free energy:

$$F(q) = \sum_z \int_\Theta q(z, \theta) \log \frac{p(D, z, \theta \mid M)}{q(z, \theta)} d\theta . \tag{6}$$

Also, if we subtract L(D)-F(q):

$$L(D) - F(q) = \sum_z \int_\Theta q(z, \theta) \log \frac{q(z, \theta)}{p(z, \theta \mid D, M)} d\theta = KL(q \parallel p) , \tag{7}$$

we arrive at the KL divergence, where KL(q||p) is the KL divergence between *q(z,θ)* and $p(z, \theta \mid D, M)$.

This means that maximizing F(q) is equivalent to minimizing the KL divergence between the distributions. As such, the entire procedure of Variational Bayes can then be thought of as

maximizing F(q), this lower bound to the marginal likelihood, to minimize the KL divergence as much as possible to obtain an accurate approximation.

As stated before, variational methods typically use some kind of approximation. In this case, we can use the mean-field approximation

$$q(z,\theta) \approx q(z)q(\theta).$$

If we plug in this approximation, we arrive at

$$\begin{aligned} L(D) &= \log \sum_z \int_\Theta p(D,z,\theta \mid M) d\theta = \log \sum_z \int_\Theta q(z,\theta) \frac{p(D,z,\theta \mid M)}{q(z,\theta)} d\theta \\ &\geq F(q) = \sum_z \int_\Theta q(z,\theta) \log \frac{p(D,z,\theta \mid M)}{q(z,\theta)} d\theta \\ &\approx \sum_z \int_\Theta q(z)q(\theta) \log \frac{p(D,z,\theta \mid M)}{q(z)q(\theta)} d\theta \ . \end{aligned} \tag{8}$$

Plugging in this approximation then allows us to take the functional derivatives, letting us maximize the expression

$$q(z) \propto \exp\left(\int_\Theta q(\theta) \log p(D,z \mid \theta, M) d\theta\right),$$

$$q(\theta) \propto p(\theta \mid M) \exp\left(\sum_z q(z) \log p(D,z \mid \theta, M)\right). \tag{9}$$

This leads to two coupled iterative equations, which we can loop over until they converge. When they converge, we will have maximized our lower bound, F(q), which can then be used as a score for model selection.

5. Structural Learning

To address the problem of structural learning, we can use a greedy algorithm in place of an exhaustive search to find the optimal network. Specifically, a hill-climbing algorithm that conducts a local search using a heuristic can be implemented (*2*). Basically, we can generate a random network to represent an initial state. After that, we can conduct three local operations on this network, which include adding, deleting, or reversing an arc. After each operation, the score of the network is determined corresponding to a fitness function. The score that can be used is actually the (negative) variational free energy, which we obtained from the Variational Bayes procedure itself. When the network with the highest score after the local operations is determined, the hill-climbing algorithm moves to this network and sets this as the current state, repeating the local search. This process is repeated until the convergence.

Hill climbing, as opposed to other greedy algorithms, does not retain any information about its previous states. To address this issue, so that the algorithm does not get stuck on a local maximum, the algorithm can be repeated or rerun with other random graphs as the initial states. After that, all the generated networks can be model averaged, where high confidence arcs from the networks can be selected and included in the final network.

6. Probabilistic Logical Programming

To process data that is incomplete and uncertain, the framework of probabilistic logical programming can be used to obtain meaningful relations. It is a framework that combines three different paradigms, namely first-order logic, statistical learning, and probability. Logical programming is the process of using first order logic in the form of declarative statements or clauses to describe relationships between entities and infer their consequences. In logical programming, facts are represented as propositions. The goal or solution can be obtained starting with the negation of a proposition to be proved and deriving a contradiction by systematic deduction using the facts. Recently, there has been interest in extending logical programming to include probabilities figure 1. This approach can allow for more realistic modeling capabilities by taking into account the uncertainties in the data obtained in real world that has significant applications in machine learning (*3*).

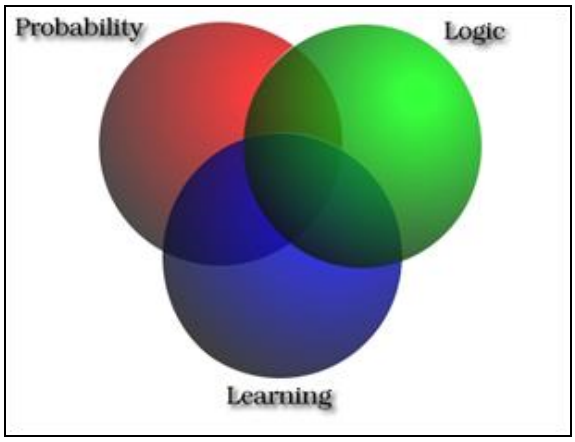

Figure 1. Schematics of probabilistic logic paradigm.

PRISM is a probabilistic logical framework based on B-prolog the language extends the Horn clauses to include random variables. The semantics of the programming language is based on distribution semantics defined on DB = {F} U {R}, where {F} is the set of atoms with random variables, called msw atoms, {R} is a set of definite clauses whose head atoms are not msw atoms, and DB is a PRISM program. The distribution semantics consists of defining a probability measure $P_F(.)$ first on the Herbrand models containing the msw atoms. Then by Kolmogorov's extension theorem the measure is extended to $P_{DB}(.)$ the product space of {F} and

{R}. In practice, PRISM allows users to easily specify discrete probability distributions through its syntax. Furthermore, it supports the use of conditional probabilities, providing the mechanisms needed to model Bayesian networks. Most importantly, it has support and implementations of some of the most popular and complex statistical learning algorithms, including ML, MAP, and Variational Bayes (*9*).

To program a Bayesian network in PRISM, one has to generate what is known as a "probabilistic clause." Basically, this is a clause that takes a random variable and generates a probabilistic call on the variables (*9*). To make the example clearer, consider a simple probabilistic model that contains three binary random variables A, B, and C. To specify the binary distribution in PRISM's syntax, one can simply use PRISM's random "switch" declaration: `values(_, [0,1]).`

Next, to model the three random variables a probabilistic call must be made to each of the variables. This is done through PRISM's version of the switch statement, known as "msw" (*9*). A simple probabilistic clause named "world" that considers the three variables would be:

world(A,B,C):- msw(a,A),msw(b,B),msw(c,C).

Now that a probabilistic model can be created, generating a Bayesian network simply requires one addition: conditional probabilities. This is easily implemented using the prolog/PRISM complex functor syntax, which is a term with parenthesis containing another term (*9*).

So consider a simple Bayesian network with three binary random variables, A, B, and C, and let us say the graph consists of an edge from A to C and an edge from B to C. A probabilistic predicate to model this would be:

world(A,B,C):-msw(a,A),msw(b,B),msw(c(A,B),c).

The project consists of generating such a probabilistic clause, a representation of a Bayesian network, which best models the real gene expression data (*4*). With this clause, one can easily use PRISM to conduct inference from this model.

7. Experiment

Flow cytometry data was obtained for the well-established "Raf" pathway in a discretized format into three levels using one-dimensional K-means.

The random generation method was implemented where two nodes are randomly selected and an edge is added on the condition that the graph remains acyclic. This process is repeated and the algorithm outputs a random graph that can be used as the initial state.

After that, the hill-climbing algorithm was implemented, where a local search was performed on the graph. The three operations, which are add, delete, and reverse, were conducted on pairs of nodes to generate the new graphs that comprised the local search space. The acyclic condition was checked for each of these three operations to generate failure-free clauses.

For each operation, the variational free energy was calculated to be used as a score. During the course of the algorithm, the graph was stored as a list of edges, i.e., [edge(pKC,pKA), edge(raf,mek)]. However, to obtain the free energy, the graph was then converted to a PRISM program model to take advantage of PRISM's learning system. A probabilistic predicate, known as "world," was generated based on the graph edges and then saved to a .psm file, which was then loaded into the database. The system was then set up for VM-EM learning through PRISM's syntax, by calling learning on the hyper-parameters of a Dirichlet distribution. Learning was begun and the free energy was obtained by calling "learning_statistics."

The graph with the highest score after the local operations was selected to be the next state. This algorithm continued until the graph reached convergence.

This process was repeated on 500 random graphs to avoid getting stuck on local maxima. To speed up the program, a multi-threaded version was implemented using Java, where a call was made to PRISM through batch-execution, to run the hill-climbing algorithm on each random graph in a separate thread. The threads were run concurrently, causing the overall time of the program to decrease by a factor depending on the number of cores in the computer. It is possible to run this multi-threaded application on a supercomputer to immensely speed up the program.

Finally, to obtain the true final structure, the frequency of all the edges found in the 500 converged graphs was recorded. Edges that appeared in at least 50% of the structures, or in this case edges that had a frequency greater than 250 were selected to be included in an averaged graph. Variational Bayes was run for a final time to obtain the hyper-parameters for this structure.

To visually display the graphs, the Graphviz software was used. Specifically, the edges in the graph structure from prolog were converted to Graphviz's "dot" language and outputted to a file. The command-line dot operator from Graphviz was called within PRISM on the exported file to generate a visual of the network.

The source code for the PRISM and Java programs is provided for the reader's convenience. Directions on using the program are included in the readme.

8. Results/Discussion

Table A-1 in the appendix shows the frequency of the edges from the 500 graphs. Edges that had a frequency above 250 were selected to be included in the final averaged model. The hyper-parameters for this final structure obtained using Variational Bayes is listed in the appendix. The MAPK/Raf pathway as defined by the literature is given in figure 2, as compared to the final generated model given in figure 3.

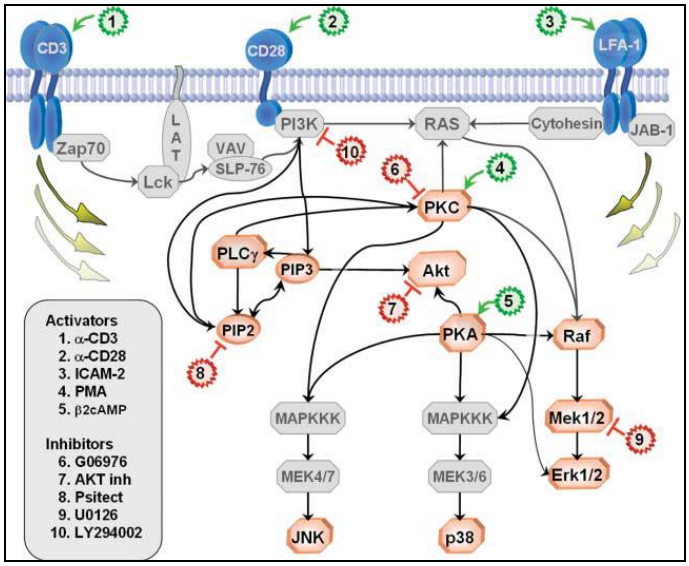

Figure 2. MAPK pathway from literature (*4*).

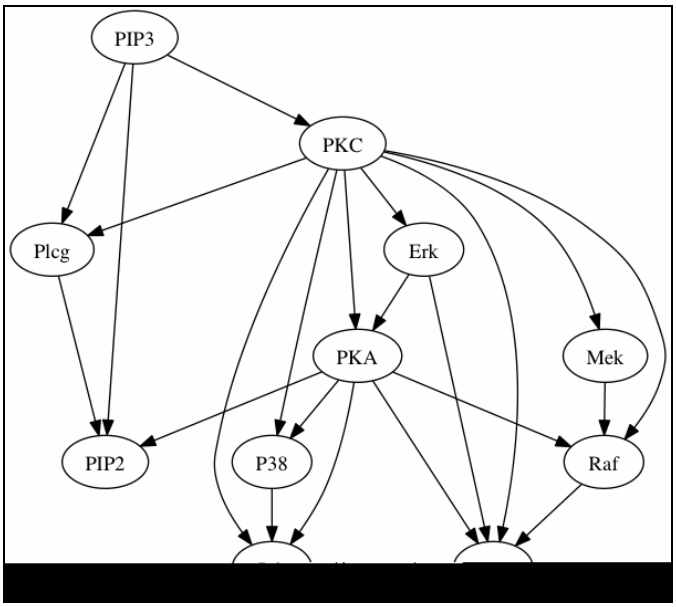

Figure 3. Generated pathway from algorithm.

The final structure contained a total of 22 edges. Each edge and its accuracy based on other models and literature is shown in table 1.

Table 1. Accuracy of edges.

Edge	Result
PKC->PKA	Expected
PKC->Raf	Expected
PKC->Mek	Expected
PKC->Erk	Expected
PKC->Akt	Expected
PKC->Plcg	Reversed
PKC->P38	Expected
PKC->Jnk	Expected
PKA->Raf	Expected
PKA->P38	Expected
PKA->PIP2	Undefined/Misplaced
PKA->Akt	Expected
PKA->Jnk	Expected
PIP3->PKC	Misplaced
PIP3->Plcg	Expected
PIP3->PIP2	Expected
Erk->PKA	Reversed
Erk->Akt	Expected
Mek->Raf	Reversed
Raf->Akt	Expected
P38->Jnk	Undefined
Plcg->PIP2	Expected
Mek->Erk	Missed

Out of the 22 edges, 17 are well defined in the literature, indicating the generated model was relatively accurate. However, the arc from PKA->Erk was reversed in the structure. This presumably caused the arc from Mek->Erk to be missed, as adding this connection would violate the acyclic condition. The arc Mek->Raf was also reversed.

When considering the phospholipids PIP2 and PIP3, it is important to recognize that there is a feedback mechanism between them. That is, the connection between PIP2 to PIP3 is direct both ways in the literature: PIP3<->PIP2. However, the Bayesian network can only handle acyclic arcs, so the arc was pointed in just one direction from PIP3->PIP2, a similar result obtained by earlier studies (*4*). Also, this feedback mechanism made it difficult to determine where to place the connection between the phospholipids to PKC, so while the literature defines the connection as PIP2->PKC, the model found PIP3->PKC. The arc between Plcg->PKC was reversed as well.

Finally, additional arcs were added between P38->Jnk and PKA->PIP2, both of which are plausible but must be verified through literature and experimentation.

When comparing the results with Sachs et al. (*4*) (who conducted the original experiment), the models appear very similar. Fourteen out of seventeen arcs in Sachs et al. (*4*) results directly match the generated model. Also, Sachs et al. (*4*) reported two not well-defined connections, PKC->PKA and Erk->AKT, which were corroborated by this generated model. However, while Sachs et al. (*4*) reversed the connection between PIP3->Plcg, the generated model actually correctly identified the direction of the connection. Furthermore, while Sachs et al. (*4*) completely missed the arcs from PIP2->PKC and Plcg->PKC, the generated model found a connection from PIP3->PKC and reversed the connection to find PKC->Plcg. Sachs et al. (*4*) also did not find the connection from PIP3->Akt, while the generated model indirectly found the connection through PIP3->PKC->Akt. This might give further insight as to why the connection was added from PIP3->PKC rather than PIP2->PKC, since adding the connection from PIP3->PKC made the indirect connection PIP3->PKC->Akt possible.

When considering the results as a whole, the generated graph was remarkably similar to the literature. Only one arc, Mek->Erk, was completely missed because of the acyclic condition, and a few arcs were misplaced or reversed. Although this missed connection between Mek->Erk was found by Sachs et al. (*4*), the generated model was successfully able to identify relationships between the phospholipids that Sachs et al. (*4*) missed or reversed. This shows that the hill-climbing method can be successfully utilized as compared to the simulated annealing approach Sachs et al. (*4*) used.

9. Conclusions

The first-order probabilistic logic framework as implemented in PRISM was successfully utilized to facilitate both parameter and structural learning of Bayesian networks to model GRNs. This approach was applied to the "Raf" pathway, and a structure that is in good agreement with the literature was generated. Variational Bayes was utilized for parameter learning and to obtain an approximation to the marginal likelihood. The free energy was used as a fitness function for model selection in structural learning.

The final structure generated by the algorithm was very close, with only a few arcs missing or reversed. Both direct and indirect connections between genes were detected, indicating how Bayesian networks can adapt to hidden or unobserved variables. Furthermore, the algorithm was able to identify connections previously missed by other research. Such probabilistic reasoning can be successfully used on flow cytometry data, where a hill-climbing approach can be implemented to generate the Bayesian networks. PRISM was also found to be a flexible and

stable platform that can be used to implement powerful and applicable learning algorithms through simple syntax.

A number of techniques can be used to extend or change the project. First, a different discretization technique can be applied on the data to obtain more accurate results. Also, results between flow cytometry and microarray data can be compared. Furthermore, the base implementation of PRISM only supports discrete distributions for Bayesian networks. The framework could be extended to include continuous, perhaps linear, Gaussian distributions and attempt learning using this broader class of probability distributions. Most importantly, Bayesian networks are defined as DAGs, or contain the acyclic condition. However, it has been noted that the "Raf" pathway that was tested, as well as numerous other biological pathways, included self-feedback mechanisms. Alternative approaches can be used to address this issue, such as the implementation of dynamic Bayesian networks, whose states are temporal. Given time-series data for gene activation, this approach can be readily applied. This provides numerous opportunities for further research and investigations.

The project explored the applicability of probabilistic logical programming as a platform to analyze data and generate models for GRNs. Bayesian networks can be used to identify statistical as well as causal relations, making them valuable inference models. The PRISM framework along with prolog is a flexible language that can easily be integrated in a wide variety of "omics" data, including genomics, proteomics, and metabolomics. Most significantly, an implementation of such an algorithm can be used to analyze new and old data to help identify connections and networks that either have not been identified or have been overlooked.

Modeling GRNs using these methods has enormous applications in medication and drug delivery. Automatically generating probabilistic models from real data can help scientists explore the effect of activating/inhibiting certain genes. By identifying and simulating important proteins and pathways, researchers can use the pathways to develop highly effective medication and drug delivery techniques. Targeting specific proteins and transduction pathways can help lead to solutions for a number of diseases, including cancer, HIV/Aids, genetic disorders, etc. Further research in Bayesian approaches using probabilistic logical programming can help establish methods to automatically analyze the large amounts of available biological data and create robust statistical and probabilistic models to conduct inference.

10. References

1. Horng, T.; Oberdoerffer, S.; Rao, A. Gene Regulation and Signal Transduction in the Immune System. *Genome Biol.* **2008**, *9*, 315.
2. Ben-Gal, I. Bayesian Networks. *Encyclopedia of Statistics in Quality and Reliability.* John Wiley & Sons: New York, 2007.
3. Chen, J.; Muggleton, S.; Santos, J. Learning Probabilistic Logic Models From Probabilistic Examples. *Mach. Learn.* **2008**, *73.*
4. Sachs, K.; Perez, O.; Pe'er, D.; Laughenburger, A. D.; Nolan, P. G. Causal Protein-Signaling Networks Derived From Multiparameter Single-Cell Data. *Science* **2005**, *308*.
5. Akha, A. A. S.; Miller, R. A. Signal Transduction in the Aging Immune System. *Curr. Opin. Immunol.* **2005**, *17*, 486.
6. Calvo, J. R.; Pozo, D.; Guerrero, J. M. Functional and Molecular Characterization of VIP Receptors and Signal Transduction in Human and Rodent Immune Systems. *Adv. Neuroimmunol.* **1996**, *6*, 39.
7. McCubrey et al. Roles of the Raf/MEK/ERK Pathway in Cell Growth, Malignant Transformation and Drug Resistance. *Biochim Biophys Acta* **2007**, *1773*, 1263.
8. Beal, M. J.; Ghahramani, Z. Variational Algorithms for Approximate Bayesian Inference. Ph.D. Thesis, Gatsby Computational Neuroscience Unit, University College, London, 2003.
9. Sato, T.; Zhou, N.; Kameya, Y.; Izumi, Y. *PRISM User's Manual*, 2009.

Appendix. Dirichlet Posteriors

PRISM is a framework that supports statistical learning of parameters that would correspond to a family of probability distributions to model given data. This is done by assuming a set of priors for the Bayesian network and outputting a set of posteriors after processing the data. At the end of a successful execution of the learning paradigm after processing the evidence data, PRISM outputs the hyper-parameters and the ML edges connecting the nodes of the Bayesian network. The frequencies of the edges and the corresponding hyper-parameters from the successful run are listed in table A-1.

Table A-1. Frequency of edges from 500 random graphs.

Frequency	Edge	Frequency	Edge	Frequency	Edge
384	PKC->Jnk	44	Plcg->Erk	10	Erk->Jnk
383	PKC->Raf	38	P38->Erk	10	Akt->P38
383	PKC->Akt	37	PIP3->P38	9	PIP2->Jnk
355	PKC->PKA	37	Jnk->PIP3	9	Jnk->PIP2
347	PKA->P38	36	Plcg->PKA	8	PIP2->Raf
344	PKC->Mek	36	Mek->Erk	8	PIP2->Mek
330	Plcg->PIP2	33	Plcg->P38	7	Raf->Plcg
329	Erk->PKA	33	Mek->PKA	7	Raf->Jnk
327	Raf->Akt	30	PKA->Plcg	6	PIP3->Jnk
326	PKA->Raf	30	Erk->Plcg	5	Akt->Plcg
320	PKC->P38	26	Mek->Plcg	4	PIP2->P38
311	PKC->Erk	26	Erk->PIP2	3	P38->Raf
311	PIP3->PIP2	25	Plcg->Akt	2	Jnk->Raf
310	PKA->Jnk	25	PIP2->PKC	1	Raf->P38
308	P38->Jnk	23	Plcg->Mek		
304	PIP3->Plcg	22	PIP3->Akt		
286	PKC->Plcg	22	Jnk->Erk		
283	PKA->Akt	21	Jnk->Plcg		
276	PKA->PIP2	21	Akt->Mek		
271	Mek->Raf	20	PIP3->Raf		
264	PIP3->PKC	20	PIP2->Erk		
255	Erk->Akt	19	Mek->P38		
244	Akt->Erk	19	Erk->Mek		
229	PKC->PIP3	18	Erk->Raf		
228	Raf->Mek	17	Raf->Erk		
216	Akt->PKA	17	Mek->Jnk		
195	Plcg->PIP3	17	Jnk->PKA		
191	Jnk->P38	16	PKA->Mek		
188	PIP2->PIP3	16	PIP2->Akt		
176	PKC->PIP2	16	P38->Mek		
175	Erk->PKC	16	Erk->P38		
173	Raf->PKA	15	P38->Plcg		
172	Plcg->PKC	15	Jnk->Mek		
170	PKA->Erk	15	Jnk->Akt		
169	PIP2->Plcg	14	Raf->PIP3		
129	PKA->PKC	14	Plcg->Jnk		
110	P38->PKA	14	PIP2->PKA		
100	Raf->PKC	13	PIP3->Erk		
83	Mek->PKC	13	P38->Akt		
78	P38->PKC	12	Akt->PIP2		
77	Erk->PIP3	11	PIP3->Mek		
70	Akt->Raf	11	Mek->PIP2		
64	PKA->PIP3	11	Mek->Akt		
62	Akt->PKC	11	Akt->Jnk		
61	P38->PIP3	10	Raf->PIP2		
59	Mek->PIP3	10	Plcg->Raf		
56	Akt->PIP3	10	PIP3->PKA		
51	Jnk->PKC	10	P38->PIP2		

Hyper-parameters for Final Structure:

```
switch(akt(3,3,2,2),unfixed,[1,2,3],[2.104781072191722e+01,1.302414590884339e+01,2.129136866551005e+00]).
switch(akt(3,2,2,2),unfixed,[1,2,3],[2.702103495541760e+01,2.610940210600353e+01,7.648154374561678e-03]).
switch(akt(3,1,2,2),unfixed,[1,2,3],[8.904400206989899e+01,5.038733461992049e+00,1.661432285509123e-02]).
switch(akt(3,1,2,1),unfixed,[1,2,3],[5.686115457087260e-02,1.495745678542937e-02,1.008150350524013e+01]).
switch(akt(3,1,1,1),unfixed,[1,2,3],[1.909226234055026e+01,2.001003736795894e+01,5.033884889921850e+00]).
switch(akt(2,3,3,3),unfixed,[1,2,3],[9.544254268681751e-02,4.010649754500932e+00,2.063750589049270e+00]).
switch(akt(2,3,3,2),unfixed,[1,2,3],[6.050761168370066e+00,3.501144129192212e+01,1.206650284576236e+01]).
switch(akt(2,3,2,3),unfixed,[1,2,3],[2.023651088929181e+00,2.300974528871280e-02,3.527255964274190e-02]).
switch(akt(2,3,2,2),unfixed,[1,2,3],[1.650277613476413e+02,4.603671943101045e+01,4.906807508632433e+01]).
switch(akt(2,2,3,3),unfixed,[1,2,3],[8.031390009854023e-02,1.042941156924351e+00,2.775819529603951e-02]).
switch(akt(2,2,3,2),unfixed,[1,2,3],[7.635126078280541e-02,1.001450362448501e+01,1.029750756933185e+00]).
switch(akt(2,2,2,3),unfixed,[1,2,3],[1.508114445380454e+01,6.139717993942639e-04,1.012409764263014e+00]).
switch(akt(2,2,2,2),unfixed,[1,2,3],[3.620358814443809e+02,5.003708448000258e+01,9.701399397950583e+01]).
switch(akt(2,1,3,1),unfixed,[1,2,3],[7.587027867548279e-03,1.325605952552555e-02,1.157067997461093e+00]).
switch(akt(2,1,2,3),unfixed,[1,2,3],[1.030104757437537e+00,4.986257948555739e-02,8.735301999628131e-02]).
switch(akt(2,1,2,2),unfixed,[1,2,3],[1.620528975976645e+02,9.052373133566819e+00,4.044824724313071e+00]).
switch(akt(2,1,2,1),unfixed,[1,2,3],[6.322135163539344e-02,1.144056160246998e-02,3.004576601118258e+01]).
switch(akt(2,1,1,1),unfixed,[1,2,3],[6.401420666732027e+01,8.507328991582474e+01,3.402030457025943e+01]).
switch(akt(1,3,3,3),unfixed,[1,2,3],[7.629887443197281e-02,5.102241602398782e+01,4.013531999235901e+01]).
switch(akt(1,3,3,2),unfixed,[1,2,3],[3.702188237863437e+01,1.390581763968984e+02,5.901970134920614e+01]).
switch(akt(1,3,2,3),unfixed,[1,2,3],[6.200131723214648e+01,1.913649690307136e+01,1.111061150413528e-02]).
switch(akt(1,3,2,2),unfixed,[1,2,3],[8.120369838156221e+02,1.270506744139521e+02,1.450177631943929e+02]).
switch(akt(1,2,3,3),unfixed,[1,2,3],[2.762721948184277e-02,2.100642249662979e+01,6.006052095909494e+00]).
switch(akt(1,2,3,2),unfixed,[1,2,3],[5.087355558004219e+00,1.517886338460945e+01,5.103631814113593e+00]).
switch(akt(1,2,2,3),unfixed,[1,2,3],[3.580371737611946e+02,1.506818532038199e+01,4.221181654750450e-02]).
switch(akt(1,2,2,2),unfixed,[1,2,3],[1.158083907679590e+03,6.204915110131702e+01,2.801120823110234e+02]).
switch(akt(1,1,3,1),unfixed,[1,2,3],[2.878554788046239e-02,1.093109810252222e-01,1.031329958006125e+00]).
switch(akt(1,1,2,3),unfixed,[1,2,3],[8.011623813061377e+00,4.304486356509662e-02,1.663247993759054e-03]).
switch(akt(1,1,2,2),unfixed,[1,2,3],[1.340004012397598e+02,4.035511080561404e+00,2.106812918324229e+01]).
switch(akt(1,1,2,1),unfixed,[1,2,3],[7.983588400113573e-02,3.946743798642527e-02,6.701914698956455e+01]).
switch(akt(1,1,1,1),unfixed,[1,2,3],[1.000302912475963e+02,1.260116025557360e+02,3.807539969641256e+01]).
switch(raf(3,2,3),unfixed,[1,2,3],[6.653632520297448e-02,3.129486203435872e-02,1.056938302610241e+00]).
switch(raf(3,2,2),unfixed,[1,2,3],[1.276925111159146e-01,2.551826935791457e-02,1.066266364755704e+00]).
switch(raf(3,1,1),unfixed,[1,2,3],[9.177258184164905e-02,2.202788392653714e+01,5.404702033917637e+01]).
switch(raf(2,3,3),unfixed,[1,2,3],[1.013878502919621e+00,1.066833475670973e+00,1.956089773097234e-01]).
switch(raf(2,3,2),unfixed,[1,2,3],[1.019824006764122e+00,2.034032489419806e+00,8.375180461278764e-02]).
switch(raf(2,2,3),unfixed,[1,2,3],[3.351641677917283e-02,1.807959689063284e+01,2.027938715904647e+00]).
switch(raf(2,2,2),unfixed,[1,2,3],[3.091634980616806e+00,5.805835814402766e+01,1.018739776632871e+00]).
switch(raf(2,2,1),unfixed,[1,2,3],[1.414132590104125e-01,2.044005778014749e+00,7.796811692921679e-04]).
switch(raf(2,1,1),unfixed,[1,2,3],[1.310066539887782e+02,1.920556235187734e+02,5.595885558280189e-02]).
switch(raf(1,3,3),unfixed,[1,2,3],[1.710158398816957e+02,7.005434752837283e+00,8.915524787266715e-02]).
switch(raf(1,3,2),unfixed,[1,2,3],[3.990690077941641e+02,1.506741231598041e+01,1.731682980632976e-02]).
switch(raf(1,3,1),unfixed,[1,2,3],[8.096261227547782e+00,1.098506550842028e+00,1.019592947714285e-01]).
switch(raf(1,2,3),unfixed,[1,2,3],[1.319073469920082e+03,2.950569849716813e+02,3.304022587047738e+01]).
switch(raf(1,2,2),unfixed,[1,2,3],[1.522038786515532e+03,4.620186580890551e+02,5.100152112356864e+01]).
switch(raf(1,2,1),unfixed,[1,2,3],[1.590841679129034e+02,1.730555243963606e+02,9.404256993046322e+01]).
switch(raf(1,1,1),unfixed,[1,2,3],[2.010560157273320e+02,2.404199771431847e-02,1.066541353855648e-01]).
switch(pIP2(3,2,3),unfixed,[1,2,3],[4.001209893813110e+00,1.703542806927804e+01,4.806430607726624e+01]).
switch(pIP2(3,2,2),unfixed,[1,2,3],[1.910316248831325e+01,1.914372839164787e+01,1.106829446665312e+01]).
switch(pIP2(3,1,3),unfixed,[1,2,3],[3.405586949746278e+01,2.703795984707132e+01,4.902293919160907e+01]).
switch(pIP2(3,1,2),unfixed,[1,2,3],[1.092005458339969e+03,4.280639217816594e+02,1.000306452511782e+02]).
switch(pIP2(2,3,3),unfixed,[1,2,3],[5.303795067493677e-03,6.170360632643512e-02,2.121154199468894e+00]).
switch(pIP2(2,2,3),unfixed,[1,2,3],[2.065960569795757e+00,3.030570590712228e+00,5.204677660508049e+01]).
switch(pIP2(2,2,2),unfixed,[1,2,3],[1.307429095748783e+01,3.710093936292090e+01,2.009253234158498e+01]).
switch(pIP2(2,1,3),unfixed,[1,2,3],[5.707735424846452e+01,3.608330938023343e+01,7.300007524174188e+01]).
switch(pIP2(2,1,2),unfixed,[1,2,3],[1.563052622036350e+03,5.500583785593574e+02,1.070333210330277e+02]).
switch(pIP2(1,3,3),unfixed,[1,2,3],[5.189687704245216e-03,1.390212089104614e-01,2.041364466879335e+00]).
switch(pIP2(1,3,2),unfixed,[1,2,3],[9.423444990123153e-03,2.695017999057225e-02,2.012154162378584e+01]).
switch(pIP2(1,3,1),unfixed,[1,2,3],[4.246389706613352e-02,4.139422989063424e-02,1.000450574320960e+02]).
switch(pIP2(1,2,3),unfixed,[1,2,3],[1.043935900938110e+00,1.112492642073998e+00,1.013314544591726e+00]).
switch(pIP2(1,2,2),unfixed,[1,2,3],[5.040965830325139e-02,3.041429338991394e+00,1.034337872165890e+00]).
switch(pIP2(1,2,1),unfixed,[1,2,3],[1.001347837479365e-01,2.230195846083114e+02,3.902331965836363e+01]).
switch(pIP2(1,1,3),unfixed,[1,2,3],[1.008582976980839e+00,7.048285773743402e+00,7.144418272563652e+00]).
switch(pIP2(1,1,2),unfixed,[1,2,3],[3.460379159040597e+02,6.103609406415170e+01,1.204744426000652e+01]).
switch(pIP2(1,1,1),unfixed,[1,2,3],[1.290411945843202e+02,8.309607521213042e+01,7.023195557334350e-02]).
switch(jnk(3,3,2),unfixed,[1,2,3],[8.170430454994460e-02,1.024688439439217e+00,2.374367836317726e-03]).
switch(jnk(3,2,3),unfixed,[1,2,3],[3.711132573641532e+01,8.399533624281141e-02,7.386758653356984e-02]).
switch(jnk(3,2,2),unfixed,[1,2,3],[1.013199116552721e+00,1.000103868249331e+00,1.134664367181037e-02]).
switch(jnk(3,1,1),unfixed,[1,2,3],[1.056438165848989e+00,1.406797633208276e+01,8.602382886156428e+01]).
switch(jnk(2,3,3),unfixed,[1,2,3],[2.114473712215470e+00,7.673272499815553e-02,2.009462138228633e+00]).
switch(jnk(2,3,2),unfixed,[1,2,3],[6.072286807279122e+00,9.029560057548634e+00,1.007838490508448e+01]).
switch(jnk(2,3,1),unfixed,[1,2,3],[1.084375659091877e+00,5.107310505392459e-02,3.284847657072554e-02]).
switch(jnk(2,2,3),unfixed,[1,2,3],[9.401724721402879e+01,6.042479771445067e+00,2.105155336122249e+00]).
switch(jnk(2,2,2),unfixed,[1,2,3],[1.007314624160750e+01,1.509462686207449e+01,6.042423106016300e+00]).
switch(jnk(2,2,1),unfixed,[1,2,3],[1.006529613207685e+00,7.298895301187280e-02,1.097588946130201e-01]).
switch(jnk(2,1,1),unfixed,[1,2,3],[1.150666444507875e+02,1.621057906790526e+02,4.409114920385093e+01]).
switch(jnk(1,3,3),unfixed,[1,2,3],[1.190916815111105e+02,5.506560385800171e+01,2.091981171398936e+00]).
switch(jnk(1,3,2),unfixed,[1,2,3],[2.720047760154043e+02,1.170000599731376e+02,2.091598517261623e+00]).
```

```
switch(jnk(1,3,1),unfixed,[1,2,3],[8.077910048079742e+00,5.354264384151852e-02,3.520228501363176e-02]).
switch(jnk(1,2,3),unfixed,[1,2,3],[1.340109193565301e+03,1.802048611547423e+02,9.098326077897324e+00]).
switch(jnk(1,2,2),unfixed,[1,2,3],[1.702085013619618e+03,3.380012366146435e+02,2.505470390984796e+01]).
switch(jnk(1,2,1),unfixed,[1,2,3],[3.781036522854555e+02,4.601735814093122e+01,3.024419914039737e+00]).
switch(jnk(1,1,1),unfixed,[1,2,3],[1.120108347300325e+02,3.806112270305465e+01,2.801085501655414e+01]).
switch(plcg(3,3),unfixed,[1,2,3],[4.062373930439295e+00,3.121674374903056e+01,4.615050780152785e-03]).
switch(plcg(3,2),unfixed,[1,2,3],[2.850503906638140e+02,9.813963657928241e+01,2.179180632025527e+00]).
switch(plcg(3,1),unfixed,[1,2,3],[2.085593842284205e+00,4.365187122098946e-02,2.035541361892429e+00]).
switch(plcg(2,3),unfixed,[1,2,3],[5.421193069092512e+02,2.911033672775044e+01,7.712251703199735e-02]).
switch(plcg(2,2),unfixed,[1,2,3],[3.716029066902827e+03,9.303882248688838e+01,7.292427984205729e-02]).
switch(plcg(2,1),unfixed,[1,2,3],[1.046719146284044e+00,1.125162630332833e+00,2.009449562128575e+01]).
switch(plcg(1,1),unfixed,[1,2,3],[2.122082265166680e+02,2.620912551646054e+02,1.000006297949995e+02]).
switch(pKA(3,3),unfixed,[1,2,3],[1.446217272069794e-02,2.800429836280441e+01,9.706578321660727e+01]).
switch(pKA(3,2),unfixed,[1,2,3],[1.401800448152959e-01,3.607834454875815e+01,2.880342186161895e+02]).
switch(pKA(3,1),unfixed,[1,2,3],[2.038690994437913e+00,3.579598885197188e-02,3.138774652535936e-02]).
switch(pKA(2,3),unfixed,[1,2,3],[9.000996203824691e+00,3.890280244728064e+02,8.303463308550330e+01]).
switch(pKA(2,2),unfixed,[1,2,3],[4.280079053433236e+02,2.062027591897890e+03,1.380068991726592e+03]).
switch(pKA(2,1),unfixed,[1,2,3],[1.070401176521088e+02,9.933643024076355e-02,7.347088576956984e-02]).
switch(pKA(1,1),unfixed,[1,2,3],[4.910394218881192e+02,2.392238953294190e-02,7.119191878955555e-02]).
switch(p38(3,3),unfixed,[1,2,3],[1.761637573424001e+02,4.130919849775311e+00,3.685935906515314e-02]).
switch(p38(3,2),unfixed,[1,2,3],[3.910580430640798e+02,2.503528874496706e+01,1.014357847346842e+00]).
switch(p38(3,1),unfixed,[1,2,3],[8.145374339507320e+00,1.086558533401370e+00,1.059189294959750e-01]).
switch(p38(2,3),unfixed,[1,2,3],[1.529061531429524e+03,1.020754013808111e+02,3.706418447213304e+01]).
switch(p38(2,2),unfixed,[1,2,3],[2.065007893683377e+03,3.101627690104216e+01,2.025722854421660e+00]).
switch(p38(2,1),unfixed,[1,2,3],[4.270206724679193e+02,1.048756975187821e+00,1.234923852097669e-02]).
switch(p38(1,1),unfixed,[1,2,3],[1.780089094066029e+02,3.211590368828645e+02,1.010020740993941e+02]).
switch(pKC(3),unfixed,[1,2,3],[4.000115967193832e+00,3.850771315893066e+02,3.514097733989625e+01]).
switch(pKC(2),unfixed,[1,2,3],[2.202115549298599e+01,3.809047177502779e+03,5.710829507413882e+02]).
switch(pKC(1),unfixed,[1,2,3],[5.740963616636188e+02,3.886825238351088e-02,8.783216870598354e-02]).
switch(mek(3),unfixed,[1,2,3],[6.010545187647992e+02,5.002412891691504e+00,2.932344195904668e-02]).
switch(mek(2),unfixed,[1,2,3],[4.108064390408638e+03,8.413150178526675e+01,2.105809311403344e+00]).
switch(mek(1),unfixed,[1,2,3],[2.010915450673237e+02,3.230023848812932e+02,7.607755204783606e+01]).
switch(erk(3),unfixed,[1,2,3],[2.753582588953640e-02,4.810623867503612e+02,1.250646310401586e+02]).
switch(erk(2),unfixed,[1,2,3],[1.346505475383242e-01,3.870092554703287e+03,3.241480586358680e+02]).
switch(erk(1),unfixed,[1,2,3],[4.910011082952420e+02,1.070252696126061e+02,2.092071430175520e+00]).
switch(pIP3,unfixed,[1,2,3],[5.740464018132127e+02,4.402011909460151e+03,4.240664767051123e+02]).
```

List of Symbols, Abbreviations, and Acronyms

AP	maximum a posteriori
DAGs	directed acyclic graphs
DNA	deoxy ribonucleic acid
GRNs	Genetic Regulatory Networks
MAP	maximum a posteriori
ML	maximum likelihood
mRNA	messenger ribooxy nucleic acid
VB-EM	Variational Bayes-Expectation Maximization

NO. OF
COPIES ORGANIZATION

1 (PDF only) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 DIRECTOR
US ARMY RESEARCH LAB
IMAL HRA
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RESEARCH LAB
RDRL CIO LL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 THOMAS JEFFERSON HIGH SCHOOL
FOR SCI AND TECHLGY
N KISHORE
6560 BRADDOCK RD
ALEXANDRIA VA 22312

ABERDEEN PROVING GROUND

2 DIR USARL
RDRL WM
S KARNA
RDRL WMM A
R BALU